

BOPF Queries

SAP AG, 2012

The SAP logo is located in the bottom left corner of the slide. It consists of the letters 'SAP' in a bold, white, sans-serif font, set against a blue rectangular background that has a slight gradient and a white diagonal line.

Disclaimer

This presentation outlines our general product direction and should not be relied on in making a purchase decision. This presentation is not subject to your license agreement or any other agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or to develop or release any functionality mentioned in this presentation. This presentation and SAP's strategy and possible future developments are subject to change and may be changed by SAP at any time for any reason without notice. This document is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. SAP assumes no responsibility for errors or omissions in this document, except if such damages were caused by SAP intentionally or grossly negligent.

Agenda

Introduction

Creating a Query

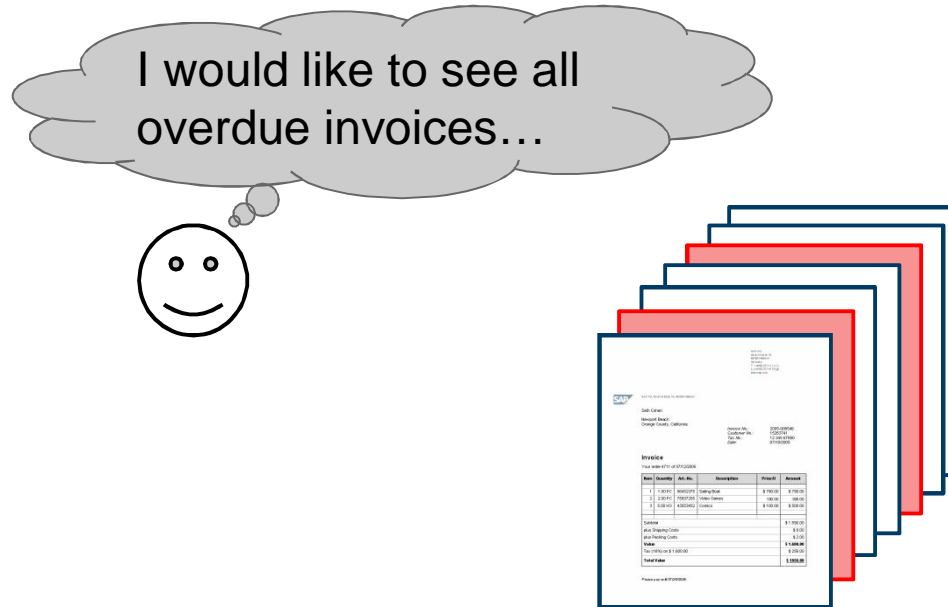
Implementing a Custom Query



Introduction

Introduction

Motivation



The application for editing customer invoices must offer a screen for displaying all overdue invoices. This is implemented using of a query located in the `CUSTOMER_INVOICE` business object that returns the invoices in question .

Introduction

Definition

A query is a business object entity and is always assigned to a certain node, whose instances are returned.

The consumer is able to query either only the keys or the whole data of the resulting node instances.

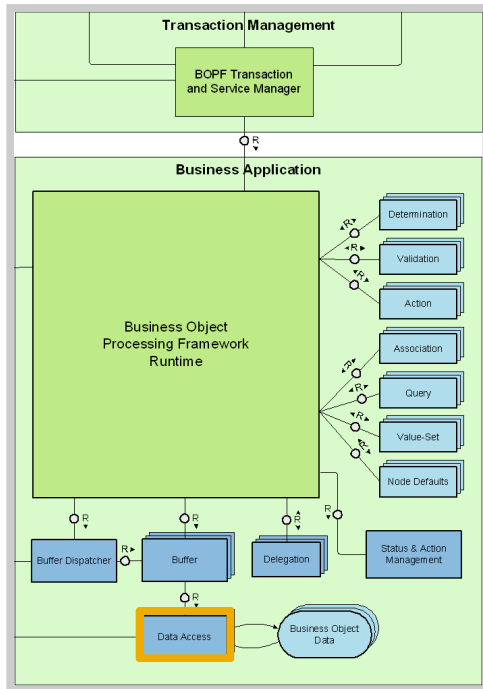
Queries never modify any node instance data.

There are three types of queries:

- Node attribute query
- Custom query
- Generic result query

Query Types

Node Attribute Queries



Resolved by BOPF DAC without any implementation work.

There are two in-built node attribute queries available:

SELECT_ALL:

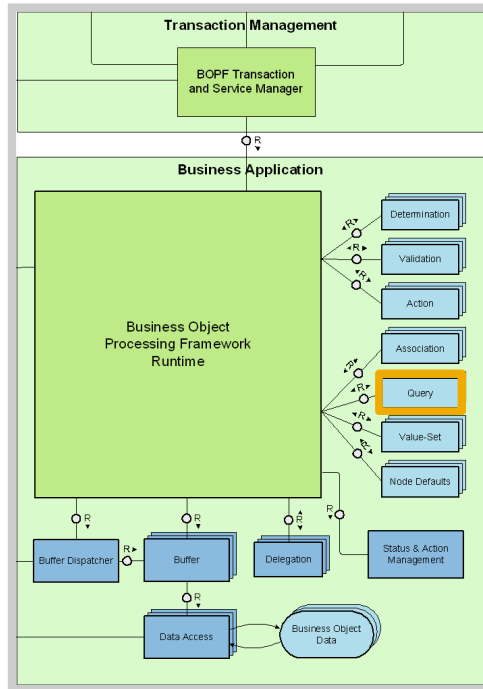
- No query parameter structure
- Returns all node instances of the assigned node of the query

SELECT_BY_ELEMENTS:

- Query parameter structure corresponds to a range table related to the assigned nodes' attributes
- Returns only node instances of the assigned node whose attributes comply with this range table

Query Types

Custom Query and Generic Result Query



Custom Query

- Needed to provide application-specific queries whose logic is not covered by node attribute queries
- Resolved using an implementing class
- Query parameter structure is arbitrary
- Returns only node instances of the assigned node of the query
- **Example:** GET_OVERDUE_INVOICES

Generic Result Query (special kind of custom query)

- Needed in rare use cases that cannot be realized with custom queries due to performance reasons (e.g. de-normalized read-only lists for the UI)
- May return any kind of data table (but these must contain a **KEY** component relating to the instances of the assigned node)



Creating a Query

Creating a Node Attribute Query

The image shows two screenshots from the SAP Business Object Detail Browser. The left screenshot shows the 'Business Object Detail Browser' for 'ZCI_CUSTOMER_INVOICE'. The 'Queries' folder is selected, and a context menu is open with 'Create Query' highlighted. The right screenshot shows the 'Maintain Business Object ZCI_CUSTOMER_INVOICE, Active Version' dialog box. The 'Query Name' is 'SELECT_BY_ELEMENTS'. The 'Node' is 'ROOT'. The 'Query Category' is 'Implemented'. The 'Query Settings' section includes 'Node' (ROOT) and 'Query Category' (Implemented). The 'Implementation' section includes 'Query Class' and 'Data Type'. The 'Query Result Types' section includes 'Result Type' and 'Result Table Type'. The 'Test Data' section includes 'Test Data Container' and 'Variant'. The 'Administrative Data' section includes 'Author', 'Creation Date', 'Last Changed By', and 'Changed On'.

To create a `SELECT_ALL` query, create a query and name it `SELECT_ALL` (or `SELECT_BY_ELEMENTS` if the query parameters are the same as the attributes of the assigned node to which the query is assigned).

Creating a Custom Query

Maintain Business Object ZCI_CUSTOMER_INVOICE, Active Version

Query Name:

Description:

Query Settings

Node	ROOT
Query Category	Implemented

Implementation

Query Class	ZCL_CI_Q_SELECT_BY_ELEMENTS
Data Type	ZCI_S_Q_SELECTION_PARAMETERS

Query Result Types

Result Type	<input type="text"/>
Result Table Type	<input type="text"/>

Test Data

Test Data Container	<input type="text"/>
Variant	<input type="text"/>

Administrative Data

Author	<input type="text"/>
Creation Date	<input type="text"/>
Last Changed By	<input type="text"/>
Changed On	<input type="text"/>

Navigation icons: [OK] [Cancel] [Back] [Forward] [Refresh]

Define the implementing class and the data type:

- The query class must implement interface `/BOBF/IF_FRW_QUERY` and contains the query logic.
- The data type can be a structure with any kind of component and can be used by a generic UI to display a range-table-based input form. At runtime, as soon as the query is executed after the form is filled, the parameter `IT_SELECTION_PARAMETERS`, which contains these values, is passed. Each line of it represents a value range related to a component contained in this data type (but not every component of the data type must always have a corresponding line in `IT_SELECTION_PARAMETERS` – passing a value range is optional). The query implementation can use the selection parameters for dynamic selections.



Query Implementation

Query Implementation

Overview

RETRIEVE_DEFAULT_PARAM
(optional)

```
methods RETRIEVE_DEFAULT_PARAM
importing
  IS_CTX type /BOBF/S_FRW_CTX_QUERY
changing
  CT_SELECTION_PARAMETERS type /BOBF/T_FRW_QUERY_SELPARAM
raising
  /BOBF/CX_FRW .
```

QUERY

```
methods QUERY
importing
  IS_CTX type /BOBF/S_FRW_CTX_QUERY
  IT_FILTER_KEY type /BOBF/T_FRW_KEY optional
  IT_SELECTION_PARAMETERS type /BOBF/T_FRW_QUERY_SELPARAM optional
  IS_QUERY_OPTIONS type /BOBF/S_FRW_QUERY_OPTIONS optional
  IO_QUERY type ref to /BOBF/IF_FRW_QUERY
  IO_READ type ref to /BOBF/IF_FRW_READ
  IO_MODIFY type ref to /BOBF/IF_FRW_MODIFY optional
exporting
  EO_MESSAGE type ref to /BOBF/IF_FRW_MESSAGE
  ET_KEY type /BOBF/T_FRW_KEY
  ES_QUERY_INFO type /BOBF/S_FRW_QUERY_INFO
  ET_DATA type index table
raising
  /BOBF/CX_FRW .
```

Queries are classes implementing the /BOBF/IF_FRW_QUERY interface.

Query Implementation

Method: RETRIEVE_DEFAULT_PARAMETERS

```
methods RETRIEVE_DEFAULT_PARAM
importing
  IS_CTX type /BOBF/S_FRW_CTX_QUERY
changing
  CT_SELECTION_PARAMETERS type /BOBF/T_FRW_QUERY_SELPARAM
raising
  /BOBF/CX_FRW .
```

Retrieves the default parameters of a query.

- **IS_CTX**: Context information of the query
- **CT_SELECTION_PARAMETERS**: Reference to the parameters of the query, which are filled with default values using this method

Query Implementation

Method: QUERY

methods QUERY

importing

IS_CTX

type /BOBF/S_FRW_CTX_QUERY

IT_FILTER_KEY

type /BOBF/T_FRW_KEY optional

IT_SELECTION_PARAMETERS

type /BOBF/T_FRW_QUERY_SELPARAM optional

IS_QUERY_OPTIONS

type /BOBF/S_FRW_QUERY_OPTIONS optional

IO_QUERY

type ref to /BOBF/IF_FRW_QUERY

IO_READ

type ref to /BOBF/IF_FRW_READ

IO_MODIFY

type ref to /BOBF/IF_FRW_MODIFY optional

exporting

EO_MESSAGE

type ref to /BOBF/IF_FRW_MESSAGE

ET_KEY

type /BOBF/T_FRW_KEY

ES_QUERY_INFO

type /BOBF/S_FRW_QUERY_INFO

ET_DATA

type index table

raising

/BOBF/CX_FRW .

Executes the query logic.

Query Implementation

Method: QUERY

IS_CTX: Context information about the query (BO key, root node key, assigned node key,...)

IT_FILTER_KEY: If the query is to respect only a restricted set of the node instances, these instances can be specified by the consumer using this parameter. If the parameter is empty, all node instances of the query's assigned node are affected by the query.

IT_SELECTION_PARAMETER: This parameter represents a range table of attributes of the query parameter structure. If attributes are supplied, only these must be filled – the other attributes do not need to be filled if this improves performance.

IS_QUERY_OPTIONS:

- **MAXIMUM_ROWS:** Maximum number of node instances returned by the query. In the case of activated paging, the number of node instances of a single page.
- **SORTING_OPTIONS:** The sorting of the results can be modified using this parameter. It consists of a table allowing the sort order to be specified (ascending/ descending) for the columns (=attributes).
- **PAGING_OPTIONS:** Allows the paging options to be modified.
- **PAGING_ACTIVE:** Yes/no
- The top instance of the current page can either be defined by row (**START_ROW**) or by instance key (**START_KEY**).

Query Implementation

Method: QUERY

IO_QUERY: Reference to an object allowing a further query to be executed.

IO_READ: Reference to an object implementing the read interface to read instance data.

IO_MODIFY: Reference to an object implementing the modifying interface to modify instance data.

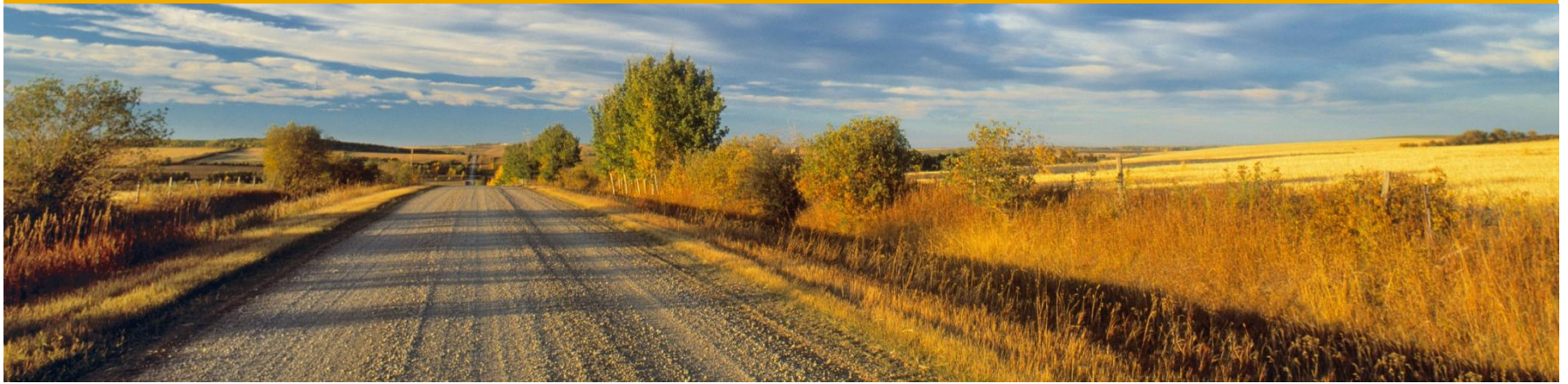
EO_MESSAGE: This message object contains all messages that are created while the query is executed.

ES_QUERY_INFO: This parameter contains a handle and the paging-related information.

- **COUNT:** Total number of results
- **QUERY_HANDLE:** A handle to identify the context

ET_DATA: Returning data table for generic result queries

ET_KEY: This parameter represents a set of the keys of the node instances that are the result of the query.



Questions & Answers

Questions

- Do `SELECT_BY_ELEMENTS` and `SELECT_ALL` queries have to be implemented?
- Can queries be assigned to nodes located in a dependent object?
- Is it possible to set properties for queries (e.g. enable/ disable)?
- Can I use queries within entity implementations (e.g. in an action)?
- If a new node instance is created during a transaction, will a query executed later on in the same transaction be able to return it?
- Queries must not change instance data, so why do I need the `io_modify` importing parameter at all?
- Is it possible to reuse other queries in the implementation of a custom query?
- Should I use the `io_read` importing parameter or an SQL statement sent to the node's database table in the implementation of a custom query?
- Is it necessary to provide the paging and sorting feature for each query?

Questions & Answers

Do `SELECT_BY_ELEMENTS` and `SELECT_ALL` queries have to be implemented?

No, if the `/BOBF/CL_DAC_TABLE` is used, these queries are resolved automatically.

Can queries be assigned to nodes located in a dependent object?

No, for instance a `SELECT_BY_ELEMENTS` query would return all instances fulfilling the search criteria. If a DO is used in multiple business objects, this result is not related to a single DO use case and thus the result makes no sense.

Is it possible to set properties for queries (e.g. enable/ disable)?

No, queries are not node instance-related, thus node category-dependent properties and dynamic properties do not cover queries.

Can I use queries within entity implementations (e.g. in an action)?

No, queries are only intended to be used by consumers or other queries. Usually, you have node instance keys provided by BOPF (`IT_KEY` importing parameter) that can be used to instances related to `retrieve/retrieve_by_associations`. In addition, you can use `convert_altern_key` to get further node instances.

Questions & Answers

If a new node instance is created during a transaction, will a query executed later on in the same transaction be able to return it?

No, queries only operate on the database image, which means that the changes done so far in the current transaction are ignored. This is a difference to `convert_altern_key` and `retrieve/retrieve_by_association`.

Queries must not change instance data, so why do I need the `io_modify` importing parameter at all?

If a custom query is assigned to a transient node, the query must write the result instances to the transient node. This ensures that the consumer is able to get them using, for instance, the `RETRIEVE` core service.

Is it possible to reuse other queries in the implementation of a custom query?

Use the importing parameter `IO_QUERY->QUERY(...)` to call a certain query located in the same BO.

Questions & Answers

Should I use the `io_read` importing parameter or an SQL statement sent to the node's database table in the implementation of a custom query?

Always use a SQL statement for performance reasons and to ensure that the query returns only instances of the database state.

Is it necessary to provide the paging and sorting feature for each query?

Yes, this is part of the query contract which must be fulfilled by each query – a consumer must be able to rely on that.



Thank you

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, PowerPoint, Silverlight, and Visual Studio are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, z10, z/VM, z/OS, OS/390, zEnterprise, PowerVM, Power Architecture, Power Systems, POWER7, POWER6+, POWER6, POWER, PowerHA, pureScale, PowerPC, BladeCenter, System Storage, Storwize, XIV, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, AIX, Intelligent Miner, WebSphere, Tivoli, Informix, and Smarter Planet are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the United States and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are trademarks or registered trademarks of Adobe Systems Incorporated in the United States and other countries.

Oracle and Java are registered trademarks of Oracle and its affiliates.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems Inc.

HTML, XML, XHTML, and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Apple, App Store, iBooks, iPad, iPhone, iPhoto, iPod, iTunes, Multi-Touch, Objective-C, Retina, Safari, Siri, and Xcode are trademarks or registered trademarks of Apple Inc.

IOS is a registered trademark of Cisco Systems Inc.

RIM, BlackBerry, BBM, BlackBerry Curve, BlackBerry Bold, BlackBerry Pearl, BlackBerry Torch, BlackBerry Storm, BlackBerry Storm2, BlackBerry PlayBook, and BlackBerry App World are trademarks or registered trademarks of Research in Motion Limited.

Google App Engine, Google Apps, Google Checkout, Google Data API, Google Maps, Google Mobile Ads, Google Mobile Updater, Google Mobile, Google Store, Google Sync, Google Updater, Google Voice, Google Mail, Gmail, YouTube, Dalvik and Android are trademarks or registered trademarks of Google Inc.

INTERMEC is a registered trademark of Intermec Technologies Corporation.

Wi-Fi is a registered trademark of Wi-Fi Alliance.

Bluetooth is a registered trademark of Bluetooth SIG Inc.

Motorola is a registered trademark of Motorola Trademark Holdings LLC.

Computop is a registered trademark of Computop Wirtschaftsinformatik GmbH.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, SAP HANA, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company.

Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase Inc. Sybase is an SAP company.

Crossgate, m@gic EDDY, B2B 360°, and B2B 360° Services are registered trademarks of Crossgate AG in Germany and other countries. Crossgate is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

The information in this document is proprietary to SAP. No part of this document may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of SAP AG.